THE KEYS TO BREAKTHROUGH APPLICATIONS

SQL パフォーマンス チューニング

インターシステムズジャパン㈱
カスタマーサポート部
サポートエンジニア
田中 歩

INTERSYSTEMS

アジェンダ

- クエリの最適化
  - テーブルチューニング
  - クエリプラン
- パフォーマンスチューニングの例
- %SYS.PTools アップデート

THE KEYS TO BREAKTHROUGH APPLICATIONS

クエリの最適化

InterSystems

最初の一歩

- テーブルチューニング
- テーブルチューニング
- テーブルチューニング！

- では、テーブルチューニングとは何か?

テーブルチューニングとは

- テーブルデータから次の2つを算出します
  - エクステントサイズ
  - 選択性 (selectivity)
- テーブル定義やデータの構造は変更しません

エクステントサイズ

- エクステントサイズ はテーブルのレコード数
  - テーブル結合を行う場合にどのテーブルから処理するかを決める手がかり
  - エクステントサイズが大きい ＝ コストが大きい

選択性 (Selectivity)

- 選択性 = フィールドのユニークデータの割合
  - "Where フィールド = <値>"
    で全体の何%のレコードを抽出するか (概数)

SQLテーブル Sample.Person のフィールド:

フィルタ: ページサイズ: 20 見つかったアイテム数: 18

カラム	データタイプ	カラム#	必須	ユニーク	照合	除し	最大長	BLOB	コンテナ	選択性	xDBCタイプ	参照先
ID	%Library.Integer	1	Yes	Yes		No		No		1	INTEGER	
Age	%Library.Integer	2	No	No		No		No		1.4925%	INTEGER	
DOB	%Library.Date	3	No	No		No		No		1.0101%	DATE	
FavoriteColors	%Library.String	4	No	No	SQLUPPER	No	50	No		2.3810%	VARCHAR	
Home	Sample.Address	5	No	No		Yes		No		1.0000%	VARCHAR	
Name	%Library.String	6	Yes	No	SQLUPPER	No	50	No		1.0000%	VARCHAR	
Office	Sample.Address	7	No	No		Yes		No		1.0000%	VARCHAR	
SSN	%Library.String	8	Yes	Yes	SQLUPPER	No	50	No		1	VARCHAR	
Spouse	%Library.Integer	9	No	No		No		No		100.0000%	INTEGER	Sample.Pe
x__classname	%Library.CacheString	10	No	No		Yes		No		100.0000%	VARCHAR	

選択性の使用

- クエリ実行時にどのインデックスを使用するかの手がかりになる→ できる限り早い段階で抽出レコード数を絞り込む
- 選択性=1 (ユニークフィールド) を優先的に使用
- それ以外のフィールドの選択性は % になる
  (値が小さいものが使われる)

内部データー構造(グローバル)

- マスターマップ

グローバル(ID)=

$LB(,フィールド1,フィールド2,フィールド3,,,)

- インデックスマップ

グローバル(インデックス名, プロパティ1,・・,ID)

=$LB(追加のデータ)

キーの値には素早くアクセスできる。

例：^PersonD(1121)=$LB("", 774, "山口太郎","ロイヤル薬品証券")

^PersonI("NameIndex", "山口太郎", 1121)=""

ある条件で検索すると?

```
...
^PersonD(1120)=$LB("", 773, "鈴木次郎","日本商社")
^PersonD(1121)=$LB("", 774, "山口太郎","ロイヤル薬品証券")
^PersonD(1122)=$LB("", 775, "佐藤三郎","関東銀行")
...
^PersonI("NameIndex", "山下一郎", 5834)=""
^PersonI("NameIndex", "山口太郎", 1121)=""
^PersonI("NameIndex", "山本花子", 1503)=""
...
```

ID=1121の人?

名前=山口太郎 さん?

条件が複数

```
...
^PersonD(1120)=$LB("", 773, "鈴木次郎","M","日本商社")
^PersonD(1121)=$LB("", 774, "山口太郎","M","ロイヤル薬品証券")
^PersonD(1122)=$LB("", 775, "佐藤三郎","M","関東銀行")
^PersonD(1503)=$LB("", 775, "佐藤三郎","F","関東銀行")
```

```
^PersonI("NameIndex", "山下一郎", 5834)=""
^PersonI("NameIndex", "山口太郎", 1121)=""
^PersonI("NameIndex", "山本花子", 1503)=""
```

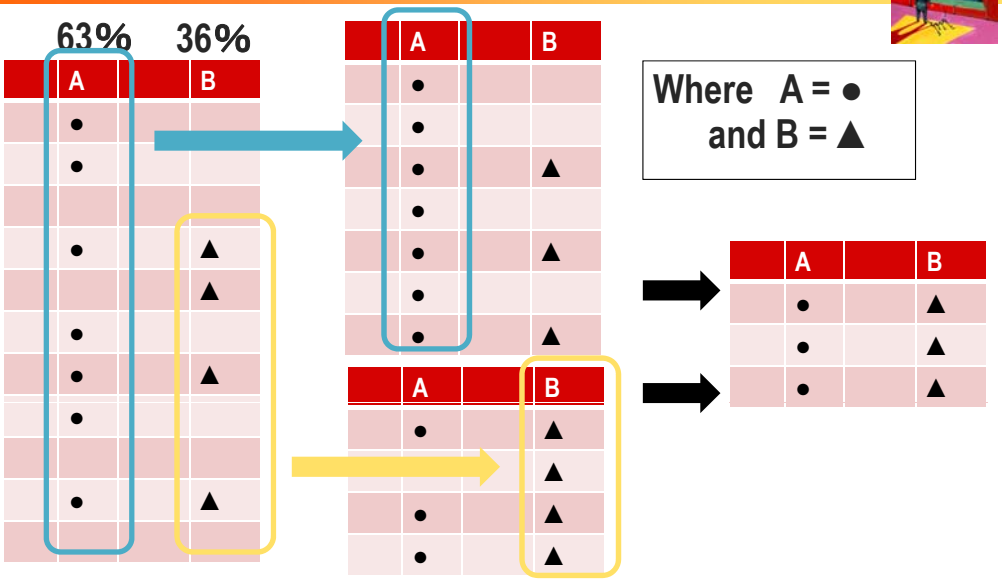
```
^PersonI("SexIndex", "M", 1120)=""
^PersonI("SexIndex", "M", 1121)=""
^PersonI("SexIndex", "M",...
^PersonI("SexIndex", "F", 1503)=""
```

名前= 山田太郎 で 性別=男性(M)
どちらのインデックス?

選択性によるインデックスの決定

- 選択性が低い ＝ そのフィールドを条件とすると対象レコードが少ない
  ↓
- より速く結果を取得できる

選択性によるインデックスの決定



13

THE KEYS TO BREAKTHROUGH APPLICATIONS

複数テーブルの結合

社員ID	氏名	年齢	部署ID
00001	山田太郎	35	3
00002	鈴木隆	28	2
00003	木村香恵	32	4
...			

1000行

<条件>

部署=営業 で
年齢 40 以上

50行

部署ID	部署名
1	総務
2	営業
3	人事
4	広報
...	

どちらのテーブルから処理?

→エクステントサイズ × 選択性

14

THE KEYS TO BREAKTHROUGH APPLICATIONS

テーブルチューニングの注意点

- すべてのテーブルで実施すること
- データの傾向が決まった後1回実行するだけでいい (データが増えても傾向が同じなら再実行は不要)
- テーブルチューニング後にクエリの実行速度を計測する

チューニングの評価

- テーブルチューニングを行った結果の評価はどうするか?
  - 実測 : 実際にSQL文を実行して所要時間を計測
    - アプリケーションレベル
    - システム管理ポータル SQL文の実行
      - パフォーマンス(秒)
      - グローバル参照数

    ただし、初回実行のクエリキャッシュ作成に注意
  - クエリプラン の 相対コスト
    - ただし同じSQLの場合のみ

クエリプランとは

- クエリがどのように実行されるか処理内容を表記したもの(英語)
  - 次ページ以降に主要なキーワードを抜粋
- 相対コスト
  - 同じクエリ(SQL)同士で比較できる相対的なコスト。主にインデックス追加前後の効果を測定できる。相対コストが低いプランのほうが良い
  - 異なるクエリ同士のコスト比較は意味がない

クエリプラン 確認方法

- 管理ポータル
  - SQL→SQL文の実行→クエリプランの表示
    (SQL文を直接入力)
  - SQL→スキーマ→クエリキャッシュ
    (実行済みクエリのプランを確認)
- スタジオでクエリを選択して右クリック→SQL文に対するプラン表示
- ターミナル do $system.SQL.Shell() の show plan コマンド (結果をテキストで残すのに便利)

クエリプラン 用語の説明

- **map (マップ)**
  - テーブルが持つデータ構造 (データ or インデックス)
  - 1つのテーブルが1つ以上のマップを持つ
- **module (モジュール)**
  - 一時テーブルを作る処理
- **temp-file (一時テーブル)**
  - ソート、結合などで必要になる一時テーブル モジュールで作成される

クエリプラン キーワード

キーワード	意味
Read master map	データグローバルを参照
Read index map	インデックスグローバルを参照
using the given yyy	主にクエリのパラメータとして与えられたyyyを使用してインデックス or データ本体の値を取得
looping on xxx	xxxでインデックス or データ本体をループ
with a %STARTSWITH range condition	前方一致条件でループ

クエリプラン キーワード(続き)

キーワード	意味
Add ID bit to bitmap temp-file A	各モジュールでの検索結果をテンポラリ領域にビットマップ形式で保存
Add a row to temp-file A, subscripted by %SQLSTRING(AAA) and ID,with node data of BBB.	各モジュールの検索結果をテンポラリ領域に サブスクリプト にAAAとIDを配列形式でデータ部にBBBを保存 例： ^temp(AAA,ID)=BBB
Accumulate the max(xxx).	xxxを計算する。Maxの場合は、比較、Sumの場合は足し算など
(((index map INDEXNAME) UNION (bitmap temp-file A)) UNION (bitmap temp-file B))	INDEXあるいはテンポラリ領域の複数の結果をUNION処理

最も悪いケース: テーブルスキャン

- **Read master map Symposia.Person.IDKEY, looping on ID.**
  - これは全データグローバルをループして参照しているということなので、データ件数の多いテーブルの場合はよくない
  - ただし、Read master map Symposia.Table1.IDKEY, using the given idkey value. は問題ない

悪いケース: 一時テーブルの作成

- Read index map Symposia.Person.NameIndex, looping on Name and ID
  Add a row to temp-file A

  一時テーブルを一時グローバルに作成している。(件数にもよるが)最適ではない。

よいケース: インデックスを使用

- Read index map Symposia.Person.NameIndex, using the given %SQLUPPER(Name) and ID

  インデックスのみを使用して結果を出力している

- (((bitmap index DatesRUs.Profile.EyeI) INTERSECT
  (bitmap index DatesRUs.Profile.GenderI)) INTERSECT
  (bitmap index DatesRUs.Profile.ActiveI))

  ビットマップインデックスのビット演算で一致するレコードを特定している

THE KEYS TO BREAKTHROUGH APPLICATIONS

パフォーマンス チューニングの例

InterSystems

#1 選択性 : 値の分布に偏り

- テーブルチューニングで選択性は算出されました。しかし…
  - レコードの 95% でそのフィールドの値がヌル
    - 実際にはほとんどレコードが抽出されないが、選択性の値は高く計算される
  - 一部の(クエリで指定されない)値がほとんどを占める
    - 95% が1つか少ない値
      - 例えば WHERE Status = 1 /* 1:完了 を示す */
    - 選択性の値が高く設定される

#1 選択性 : 対処

- SELECTIVITY を手動で設定する(低い値に)
  - スタジオ または 管理ポータルを使用
- プロパティ・パラメータ CALCSELECTIVITY = 0 を設定
  - テーブルチューニングでこのプロパティの計算を行わない

→ インデックスが利用されるようになる

#2 継承

- 1つのテーブルに対する単順なクエリでグローバル参照数が大量になる
  - SELECT Name FROM Symposia.Employee
    - 行数: 105
    - パフォーマンス: 0.050 秒
    - 80848 グローバル参照
- 何が問題か?

#2 継承 – 問題点

- SELECT Name FROM Symposia.Employee
  - 行数: 105
  - パフォーマンス: 0.050 秒
  - 80848 グローバル参照
- Class Symposia.Employee Extends Symposia.Person
  - 同じデーターマップに格納される。
    → 継承元のデータを含めてテーブルスキャンが発生

#2 継承 : 対処

- 継承順を変更してマップを分ける
  - Class Symposia.Employee Extends (%Persistent, Symposia.Person)
  - Personテーブルの参照でEmployeeを含めなくていい場合に利用可
- エクステントインデックスをサブクラスに定義
  - Index EmployeExtent [Extent, Type = bitmap];
    - 行数: 105
    - パフォーマンス: 0.007 秒
    - 851 グローバル参照

#3 マルチインデックス

- アドホッククエリなど、WHERE句条件がユーザーにより自由に設定される場合にどうインデックスを定義すればいいか?

```
SELECT ID FROM DatesRUs.Profile
WHERE Active = ?
   AND Gender = ?
   AND ( (Hair = ? and Eye = ?)
     OR  (Hair = ? and Eye = ?)  )
```


#3 マルチインデックス

- 各フィールド個別にビットマップインデックスを定義する
  - Index EyeIndex On Eye [Type = bitmap];
  - Index HairIndex On Hair [Type = bitmap];
  - Index GenderIndex On Gender [Type = bitmap];
  - Index ActiveIndex On Active [Type = bitmap];
- 標準インデックスでもよいが、ビットマップの方が速い

#3 マルチインデックス : プラン

- **Generate a stream of idkey values using the multi-index combination:**

  (((bitmap index DatesRUs.Profile.GenderIndex)
  INTERSECT (bitmap index DatesRUs.Profile.ActiveIndex))
  INTERSECT (((bitmap index DatesRUs.Profile.EyeIndex)
  INTERSECT (bitmap index DatesRUs.Profile.HairIndex))
  UNION ((bitmap index DatesRUs.Profile.EyeIndex)
  INTERSECT (bitmap index DatesRUs.Profile.HairIndex))))

#4 インデックスの照合

- 定義しているインデックスが使用されない
  - Index NameIdx on Name As SqlString;

  select * from Symposia.Collation where Name = ?

  - 相対コスト = 1600
    Read master map Symposia.Collation.IDKEY, looping on ID.
    For each row:
      Output the row.

#4 インデックスの照合 : 対処

- 照合が一致してない場合はインデックスが選択されない
  - クエリで照合をあわせる
    select * from Symposia.Collation
    where %SqlString Name = ?
  - 相対コスト = 454.4
    Read index map Symposia.Collation.IdxName,
    using the given %SQLSTRING(Name), and looping on ID.
    For each row:
    Read master map Symposia.Collation.IDKEY, using the given idkey value.
    Output the row.

#5 テスト環境 と 本番環境

- テスト環境でのクエリパフォーマンス: 問題なし
- 同じソースを使用した本番環境でクエリの実行に時間がかかる
- エクステントサイズと選択性の値を比較する
- デフォルトのエクスポートではテーブルチューニングの結果は含まれない
  - /exportselectivity=1 が必要

THE KEYS TO BREAKTHROUGH APPLICATIONS

%SYS.PTOOLS アップデート

InterSystems

SQLレベルの計測ツール

- **%SYS.PTools.SQLStats**
  - クエリの実行時間を測定し、データベースに保存する
  - 有効にすると、クエリキャッシュ内に計測用のコードが埋め込まれる
- **設定方法**
  システム全体: Do $SYSTEM.SQL.SetSQLStats(n)
  プロセス単位: Do $SYSTEM.SQL.SetSQLStatsJob(n)
  - 0：計測用コード生成を無効にする
  - 1：計測用コードを生成するが、測定は行わない
  - 2：計測用コードを生成し、クエリの開始・終了を測定する
  - 3：計測用コードを生成し、クエリの開始・終了、モジュール毎の時間も測定する

結果確認クエリ１

```
SELECT RoutineName, ModuleName, ModuleCount,
   GlobalRefs, LinesOfCode, TotalTime, RowCount,
   QueryType, StartTime, QueryText

      FROM %SYS_PTools.SQLStatsView

         WHERE Namespace= 'SYMPOSIA'
```

SQLCODE: 100 行数: 13　　パフォーマンス: 0.003 秒　183 グローバル参照

#	RoutineName	ModuleName	ModuleCount	GlobalRefs	LinesOfCode	TotalTime	RowCount	QueryType	StartTime	QueryText
1	%sqlcq.NCUH.1	MAIN	1	156	2550	.003186	16	Unknown	2011-04-26 15:50:34	SELECT * FROM Symposia . Table5 WHERE Sale
2	%sqlcq.NCUH.11	MAIN	1	6	459	.000306	0	Unknown	2011-04-26 15:54:08	SELECT RoutineName , ModuleName , ModuleCo
3	%sqlcq.NCUH.11	MAIN	1	6	408	.00029	0	Unknown	2011-04-26 15:54:12	SELECT RoutineName , ModuleName , ModuleCo
4	%sqlcq.NCUH.11	MAIN	1	147	3417	.004586	11	Unknown	2011-04-26 15:54:20	SELECT RoutineName , ModuleName , ModuleCo
5	%sqlcq.NCUH.16	MAIN	1	2027	6375	.002389	1	SELECT	2011-04-26 15:50:14	SELECT P1 FROM Symposia . Table1 WHERE P2
6	%sqlcq.NCUH.17	MAIN	1	254	2652	.00132	16	SELECT	2011-04-26 15:50:55	SELECT * FROM Symposia . Table5 WHERE { fn
7	%sqlcq.NCUH.18	MAIN	1	2120	12699	.010762	38	SELECT	2011-04-26 15:51:04	SELECT * FROM Symposia . Table1 WHERE $ EX
8	%sqlcq.NCUH.19							SELECT		SELECT RoutineName , ModuleName , ModuleCo
9	%sqlcq.NCUH.2	MAIN	1	156	2550	.00131	16	Unknown	2011-04-26 15:50:48	SELECT * FROM Symposia . Table5 WHERE Sale
10	%sqlcq.NCUH.4	MAIN	1	2423375	8380320	2.853697	1	Unknown	2011-04-26 15:49:43	SELECT COUNT (*) FROM Symposia . Table2 ,
11	%sqlcq.NCUH.5	MAIN	1	112	2448	.001299	8	Unknown	2011-04-26 15:51:28	SELECT RoutineName , ModuleName , ModuleCo
12	%sqlcq.NCUH.cls11.1							SELECT		DECLARE QRS CURSOR FOR SELECT COUNT (
13	SQL1	MAIN	1	14	102	.000186	1	SELECT	2011-04-26 15:48:52	select count(*) into :cnt from Symposia.Person

完了

結果確認クエリ２

```
SELECT RoutineName as クエリキャッシュ名,
   ModuleName as モジュール名, SUM(ModuleCount)
   AS クエリ実行回数, AVG(TotalTime) AS 平均実行時間,
   SUM(TotalTime) AS 合計実行時間, AVG(GlobalRefs)
   AS 平均グローバル参照数, AVG(LinesOfCode) AS 平均
   コード実行行数, QueryText as クエリテキスト

FROM %SYS_PTools.SQLStatsView

   WHERE NameSpace = 'SYMPOSIA'

      GROUP BY RoutineName, ModuleName

         ORDER BY 合計実行時間 DESC
```

＊クエリ単位で集計し、合計実行時間でソートすることでシステムへの影響が大きいクエリを洗い出すことができる。

%SYS.PTools 今後の予定

- システム管理ポータルから利用できる
  - 有効/無効設定の変更
  - 結果の表示

41

「InterSystems FAQ」のお知らせ

「InterSystems FAQ」サイトが新しくなりました！

キーワードでのトピック検索が可能になり
参照の多いトピックのご紹介や
関連トピックの相互参照機能などを追加いたしました。

より使いやすくなったFAQサイトを是非ご活用ください。

http://faq.intersystems.co.jp

InterSystemsでは、お客様の疑問や不明点を速やかに解消できるよう、
ウェブサイトにFAQを掲載し、定期的に内容を更新しています。

42

THE KEYS TO BREAKTHROUGH APPLICATIONS